

Machine learning: Part 7

- A logical formulation of **inductive learning**
- knowledge-based inductive learning
- Abduction and diagnosis

Motivation

- We bring together the work on knowledge representation and learning.
- Inductive learning: construct a function that has the input - output behavior observed in the data.
- We study inductive learning methods that can take advantage of prior knowledge about the world.
- In most cases, the prior knowledge is represented as general first-order logical theories.

Recall the restaurant example

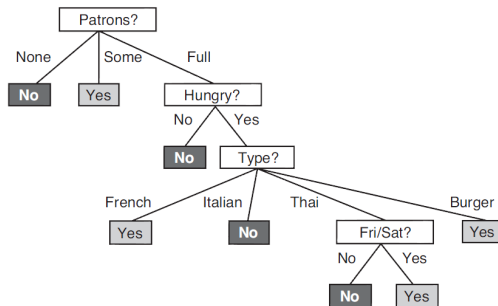
- Learning a rule for deciding whether to wait for a table.
- Examples were described by attributes

Example	Attributes										Target
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>Wait</i>
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0-10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30-60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0-10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10-30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0-10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0-10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0-10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10-30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0-10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30-60	T

A logical formulation of examples and hypotheses

- An example X_i is described by a logical sentence, e.g.
 $Alternate(X_1) \wedge \neg Bar(X_1) \wedge \neg Fri/Sat(X_1) \wedge Hungry(X_1) \wedge \dots$
- The classification of the example is given by a literal using the goal predicate, e.g., $WillWait(X_1)$ or $\neg WillWait(X_1)$
- The complete training set is expressed as the conjunction of all the example descriptions and goal literals
- Each hypothesis h_j has the form $\forall x. Goal(x) \Leftrightarrow C_j(x)$, where $C_j(x)$ is a candidate definition – some expression involving the attribute predicates.
- The extension of a hypothesis is the set of examples satisfying the definition

An example hypothesis





$$\begin{aligned}\forall r \quad WillWait(r) \Leftrightarrow & Patrons(r, Some) \\ & \vee Patrons(r, Full) \wedge Hungry(r) \wedge Type(r, French) \\ & \vee Patrons(r, Full) \wedge Hungry(r) \wedge Type(r, Thai) \\ & \quad \wedge Fri/Sat(r) \\ & \vee Patrons(r, Full) \wedge Hungry(r) \wedge Type(r, Burger) .\end{aligned}$$

Consistency of an example and an hypothesis

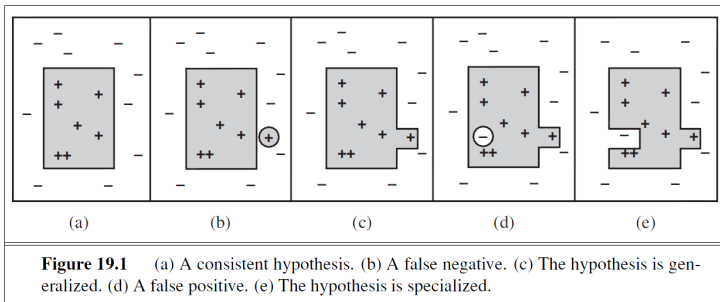
- An example can be a false negative for the hypothesis, if the hypothesis says it should be negative but in fact it is positive.
- An example can be a false positive for the hypothesis, if the hypothesis says it should be positive but in fact it is negative.
- If an example is a false positive or false negative for a hypothesis, then the example and the hypothesis are logically inconsistent with each other.

A logical formulation of inductive learning

- Suppose the hypothesis space $\mathcal{H} = \{h_1, \dots, h_n\}$.
- Initially, the learning algorithm believes $h_1 \vee \dots \vee h_n$ 
- As the examples arrive, hypotheses that are inconsistent with the examples can be ruled out. 
- Because the hypothesis space is usually vast (or even infinite), it is not feasible to build a learning system using a complete enumeration of the hypothesis space and elimination of hypotheses.
- Instead, we will describe two approaches that find logically consistent hypotheses with much less effort.

Current-best-hypothesis search

- The idea is to maintain a single hypothesis, and to adjust it as new examples arrive in order to maintain consistency.
- The basic algorithm was described by John Stuart Mill (1843), and may well have appeared even earlier.



function CURRENT-BEST-LEARNING(*examples*, *h*) **returns** a hypothesis or fail

if *examples* is empty **then**

return *h*

$e \leftarrow \text{FIRST}(\text{examples})$

if *e* is consistent with *h* **then**

return CURRENT-BEST-LEARNING(REST(*examples*), *h*)

else if *e* is a false positive for *h* **then**

for each *h'* **in** specializations of *h* consistent with *examples* seen so far **do**

$h'' \leftarrow \text{CURRENT-BEST-LEARNING}(\text{REST}(\text{examples}), h')$

if $h'' \neq \text{fail}$ **then return** h''



else if *e* is a false negative for *h* **then**

for each *h'* **in** generalizations of *h* consistent with *examples* seen so far **do**

$h'' \leftarrow \text{CURRENT-BEST-LEARNING}(\text{REST}(\text{examples}), h')$

if $h'' \neq \text{fail}$ **then return** h''



return fail

Figure 19.2 The current-best-hypothesis learning algorithm. It searches for a consistent hypothesis that fits all the examples and backtracks when no consistent specialization/generalization can be found. To start the algorithm, any hypothesis can be passed in; it will be specialized or generalized as needed.

Generalization and specialization

- We have defined generalization and specialization as operations that change the extension of a hypothesis.
- How to implement them as syntactic operations?
- Note that generalization and specialization are also logical relationships between hypotheses.
- If hypothesis h_1 with definition C_1 , is a generalization of hypothesis h_2 with definition C_2 , then $\forall x. C_2(x) \Rightarrow C_1(x)$
- e.g., if $C_2(x)$ is $Alternate(x) \wedge Patrons(x, Some)$, one possible generalization is $C_1(x) \equiv Patrons(x, Some)$. This is called dropping conditions.
- Similarly, we can specialize a hypothesis by adding extra conditions to its candidate definition or by removing disjuncts from a disjunctive definition.

The restaurant example

Example	Attributes										Target
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>Wait</i>
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0-10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30-60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0-10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10-30	T

- $h_1 : \forall x. WillWait(x) \Leftrightarrow Alternate(x)$
- $h_2 : \forall x. WillWait(x) \Leftrightarrow Alternate(x) \wedge Patrons(x, some)$
- $h_3 : \forall x. WillWait(x) \Leftrightarrow Patrons(x, some)$
- $h_4 : \forall x. WillWait(x) \Leftrightarrow Patrons(x, some) \vee Patrons(x, Full) \wedge Fri/Sat(x)$

Least-commitment search

- The current-best-hypothesis search has to do backtracking because it has to choose a hypothesis as its best guess even though it does not have enough data to be sure of the choice.
- Instead, we can keep all and only those hypotheses that are consistent with all the data so far
- Each new example will have no effect or will get rid of some of the hypotheses.
- The set of hypotheses remaining is called the version space.

function VERSION-SPACE-LEARNING(*examples*) **returns** a version space

local variables: V , the version space: the set of all hypotheses

$V \leftarrow$ the set of all hypotheses

for each example e in *examples* **do**

if V is not empty **then** $V \leftarrow$ VERSION-SPACE-UPDATE(V, e)

return V

function VERSION-SPACE-UPDATE(V, e) **returns** an updated version space

$V \leftarrow \{h \in V : h \text{ is consistent with } e\}$

Figure 19.3 The version space learning algorithm. It finds a subset of V that is consistent with all the *examples*.

Also called the candidate elimination algorithm

How to represent the enormous version space?

- A simple analogy: how do you represent all the real numbers between 1 and 2?
- Just **specify the boundaries** because we have an ordering on the real numbers.
- We also have an ordering on the version space, *i.e.*, generalization/specialization.
- This is a **partial ordering**, so each boundary will not be a point but rather a set of hypotheses called a boundary set.
- We use **two boundary sets**: a most general boundary (**the G-set**) and a most specific boundary (**the S-set**).
- Everything in between is consistent with the examples.

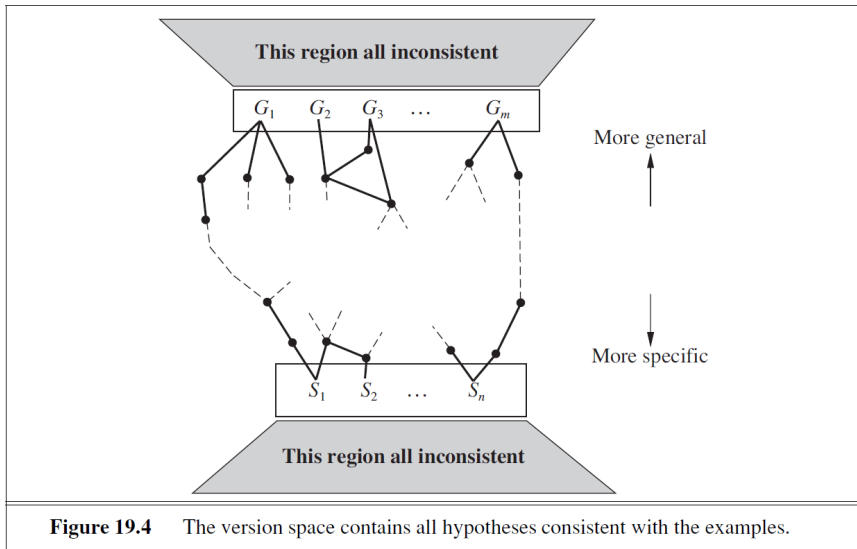
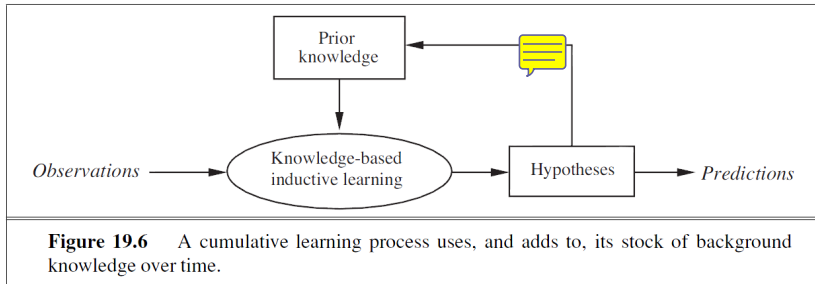


Figure 19.4 The version space contains all hypotheses consistent with the examples.

Knowledge in learning

- Let **Descriptions** denote the conjunction of all the example descriptions in the training set, and
- let **Classifications** denote the conjunction of all the example classifications.
- **Pure** inductive learning:
 $Hypothesis \wedge Descriptions \models Classifications$
- However, we would like to take advantage of prior knowledge about the world.
- Knowledge-based inductive learning:
 $\underline{Background} \wedge Hypothesis \wedge Descriptions \models Classifications$

Cumulative learning



Knowledge-based inductive learning

- Attribute-based learning methods assume that **feature values are meaningful**
- However, when the examples are about individuals and relations, **the values of properties** are not meaningful values but **names of individuals**.
- It is the properties of these individuals and their relationship to other individuals that needs to be learned.
- Knowledge-based inductive learning has been studied mainly in the field of “**Inductive Logic Programming**” as the representations are often logic programs.

Logic Programs

- A clause is a disjunction of literals
- A definite clause is a clause with exactly one positive literal
- A definite clause $H \vee \neg B_1 \vee \dots \vee \neg B_n$ is usually written in the form of a rule $H \leftarrow B_1 \wedge \dots \wedge B_n$
- H is called the head of, and $B_1 \wedge \dots \wedge B_n$ the body
- A definite logic program is a set of definite clauses

Example: trading agent

What does **Joe** like?

Individual	Property	Value
joe	likes	resort_14
joe	dislikes	resort_35
...
resort_14	type	resort
resort_14	near	beach_18
beach_18	type	beach
beach_18	covered_in	ws
ws	type	sand
ws	color	white
...

Values of properties may be meaningless names.

Example: trading agent

We are interested in learning

$$\begin{aligned} \text{prop}(\text{joe}, \text{likes}, R) \leftarrow \\ \text{prop}(R, \text{type}, \text{resort}) \wedge \\ \text{prop}(R, \text{near}, B) \wedge \\ \text{prop}(B, \text{type}, \text{beach}) \wedge \\ \text{prop}(B, \text{covered_in}, S) \wedge \\ \text{prop}(S, \text{type}, \text{sand}). \end{aligned}$$


Joe likes resorts that are near sandy beaches.

Inductive Logic Programming: Inputs and Output

- A is a set of atoms whose definitions the agent is learning.
- E^+ is a set of ground atoms observed true: positive examples
- E^- is the set of ground atoms observed to be false: negative examples
- B is a set of clauses: background knowledge
- H is a space of possible hypotheses. H can be the set of all logic programs defining A .

The aim is to find a simplest hypothesis $h \in H$ such that $B \wedge h \models E^+$ and $B \wedge h \not\models E^-$

The example

- $A = \{prop(joe, likes, R)\}$
- $E^+ = \{prop(joe, likes, resort_14), \dots\}$
-  $E^- = \{prop(joe, likes, resort_35), \dots\}$
- $B = \{prop(resort_14, type, resort),$
 $prop(resort_14, near, beach_18), \dots\}$
- H is a set of logic programs defining $prop(joe, likes, R)$.
The heads of the clauses unify with $prop(joe, likes, R)$.
 H is too big to enumerate.

Generality of Hypotheses

Hypothesis h_1 is more general than h_2 if h_1 logically implies h_2 .
 h_2 is then more specific than h_1 .

Consider the logic programs:

- $a \leftarrow b.$
- $a \leftarrow b \wedge c.$
- $a \leftarrow b. \quad a \leftarrow c.$
- $a.$

Which is the most general? Least general?

- For target relation $A = \{t(X_1, \dots, X_n)\}$ what is the most general logic program?
- What is the least general logic program that is consistent with E^+ and E^- ?

Inductive Logic Programming: Approach 1

Single target relation: $A = \{t(X_1, \dots, X_n)\}$.

- Start with the most general hypothesis and make it more complicated to fit the data.
- Most general hypothesis is $t(X_1, \dots, X_n)$.
- Keep adding conditions, ensuring it always implies the positive examples.
- At each step, exclude some negative examples.

Inductive Logic Programming: Approach 2

Single target relation: $A = \{t(X_1, \dots, X_n)\}$.

- Start with a hypothesis that fits the data and keep making it simpler while still fitting the data.
- Initially the logic program can be E^+ .
- Operators simplify the program, ensuring it fits the training examples.

ILP: General to Specific Search

Maintain a logic program G that entails the positive examples.

Initially: $G = \{t(X_1, \dots, X_n) \leftarrow\}$

A specialization operator takes G and returns set S of clauses that specializes G . Thus $G \models S$.

Three primitive specialization operators:

- Split a clause in G on condition c Clause $a \leftarrow b$ in G is replaced by two clauses: $a \leftarrow b \wedge c$ and $a \leftarrow b \wedge \neg c$.
- Split clause $a \leftarrow b$ on variable X producing:
 $a \leftarrow b \wedge X = t_1 \dots a \leftarrow b \wedge X = t_k$.
where the t_i are terms.
- Remove any clause not necessary to prove the positive examples.

Top-down ILP

```
1: procedure TDInductiveLogicProgram( $t, B, E^+, E^-, R$ )
2:    $t$ : an atom whose definition is to be learned
3:    $B$ : background knowledge is a logic program
4:    $E^+$ : positive examples
5:    $E^-$ : negative examples
6:    $R$ : set of specialization operators
7:   Output: logic program that classifies  $E^+$  positively and
    $E^-$  negatively or  $\perp$  if no program can be found
8:    $H \leftarrow \{t(X_1, \dots, X_n) \leftarrow\}$ 
9:   while there is  $e \in E^-$  such that  $B \cup H \models e$  do
10:    if there is  $r \in R$  such that  $B \cup r(H) \models E^+$  then
11:      Choose  $r \in R$  such that  $B \cup r(H) \models E^+$ 
12:       $H \leftarrow r(H)$ 
13:    else
14:      return  $\perp$ 
15: return  $H$ 
```

The example

- $H \leftarrow \{prop(joe, likes, R) \leftarrow\}$
- It could consider splitting on the property type, and keeping only those that are required to prove the positive examples.
- This results in the following clause (assuming the positive examples only include resorts):
 $\{prop(joe, likes, R) \leftarrow prop(R, type, resort)\}.$

Logical Reasoning: 3 types

Given preconditions α , post-conditions β and the rule $\alpha \rightarrow \beta$

- Deduction: determining β . It is using the rule and its preconditions to make a conclusion
- Induction: determining R1. It is learning R1 after numerous examples of β and α .
- Abduction: determining α . It is using the post-condition and the rule to assume that the precondition could explain the postcondition

Diagnosis

One simple version of diagnosis uses abductive reasoning

KB has facts about symptoms and diseases

including: $(Disease \wedge Hedges \supset Symptoms)$

Goal: find disease(s) that best explain observed symptoms

Observe: we typically do not have knowledge of the form

$(Symptom \wedge \dots \supset Disease)$

so reasoning is not deductive

Example:

$(\text{tennis-elbow} \supset \text{sore-elbow})$
 $(\text{tennis-elbow} \supset \text{tennis-player})$
 $(\text{arthritis} \wedge \text{untreated} \supset \text{sore-joints})$
 $(\text{sore-joints} \supset \text{sore-elbow} \wedge \text{sore-hip})$

Explain: sore-elbow

Want: tennis-elbow,
(arthritis \wedge untreated),
...

Non-uniqueness: multiple equally good explanations

+ logical equivalences: $(\text{untreated} \wedge \neg \text{arthritis})$

Adequacy criteria

Given KB, and β to be explained, we want an α such that

1. α is sufficient to account for β

$$KB \cup \{\alpha\} \models \beta \quad \text{or} \quad KB \models (\alpha \supset \beta)$$

2. α is not ruled out by KB

$$KB \cup \{\alpha\} \text{ is consistent} \quad \text{or} \quad KB \not\models \neg\alpha$$

otherwise $(p \wedge \neg p)$ would count as an explanation

3. α is as simple as possible

parsimonious : as few *terms* as possible
explanations should not unnecessarily
strong or unnecessarily weak

e.g. $KB = \{(p \supset q), \neg r\}$ and $\beta = q$

$\alpha = (p \wedge s \wedge \neg t)$ is too strong

$\alpha = (p \vee r)$ is too weak

4. α is in the appropriate vocabulary

atomic sentences of α should be drawn
from H, possible hypotheses in terms of
which explanations are to be phrased
e.g. diseases, original causes

e.g. sore-elbow explains sore-elbow
trivial explanation

sore-joints explains sore-elbow
may or may not be suitable

Call such α an explanation of β wrt KB

Circuit example

Components

$\text{Gate}(x) \equiv \text{Andgate}(x) \vee \text{Orgate}(x) \vee \text{Xorgate}(x)$

$\text{Andgate}(a1), \text{Andgate}(a2),$

$\text{Orgate}(o1),$

$\text{Xorgate}(b1), \text{Xorgate}(b2)$

$\text{Fulladder}(f)$ the whole circuit

Connectivity

$\text{in1}(b1) = \text{in1}(f), \text{in2}(b1) = \text{in2}(f)$

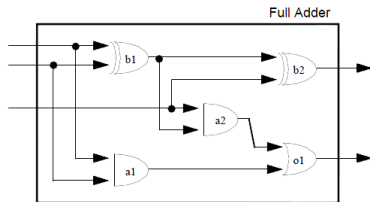
$\text{in1}(b2) = \text{out}(b1), \text{in2}(b2) = \text{in3}(f)$

$\text{in1}(a1) = \text{in1}(f), \text{in2}(a1) = \text{in2}(f)$

$\text{in1}(a2) = \text{in3}(f), \text{in2}(a2) = \text{out}(b1)$

$\text{in1}(o1) = \text{out}(a2), \text{in2}(o1) = \text{out}(a1)$

$\text{out1}(f) = \text{out}(b2), \text{out2}(f) = \text{out}(o1)$



Circuit behaviour

Truth tables for logical gates

$\text{and}(0,0) = 0, \text{ and}(0,1) = 0, \dots$ $\text{or}(0,0) = 0, \text{ or}(0,1) = 1, \dots$
 $\text{xor}(0,0) = 0, \text{ xor}(0,1) = 1, \dots$

Normal behaviour

$\text{Andgate}(x) \wedge \neg \text{Ab}(x) \supset \text{out}(x) = \text{and}(\text{in1}(x), \text{in2}(x))$
 $\text{Orgate}(x) \wedge \neg \text{Ab}(x) \supset \text{out}(x) = \text{or}(\text{in1}(x), \text{in2}(x))$
 $\text{Xorgate}(x) \wedge \neg \text{Ab}(x) \supset \text{out}(x) = \text{xor}(\text{in1}(x), \text{in2}(x))$

Abnormal behaviour: fault models

Examples

$[\text{Orgate}(x) \vee \text{Xorgate}(x)] \wedge \text{Ab}(x) \supset \text{out}(x) = \text{in2}(x)$ (short circuit)

Other possibilities ...

- some abnormal behaviours may be inexplicable
- some may be compatible with normal behaviour on certain inputs

Abductive diagnosis

Given KB as above + input settings

$$\text{e.g. } \text{KB} \cup \{\text{in1}(f) = 1, \text{in2}(f) = 0, \text{in3}(f) = 1\}$$

we want to explain observations at outputs

$$\text{e.g. } (\text{out1}(f) = 1 \wedge \text{out2}(f) = 0)$$

in the language of Ab

We want conjunction of Ab literals α such that

$$\text{KB} \cup \text{Settings} \cup \{\alpha\} \models \text{Observations}$$

Compute by “propositionalizing”:

For the above, x ranges over 5 components and u, v range over 0 and 1.

Easiest to do by preparing a table ranging over all Ab literals, and seeing which conjunctions entail the observations.

Table for abductive diagnosis

	Ab(b1)	Ab(b2)	Ab(a1)	Ab(a2)	Ab(o1)	Entails observation?
1.	Y	Y	Y	Y	Y	N
2.	Y	Y	Y	Y	N	N
3.	Y	Y	Y	N	Y	N
4.	Y	Y	Y	N	N	N
5.	Y	Y	N	Y	Y	Y
6.	Y	Y	N	Y	N	N
7.	Y	Y	N	N	Y	Y
8.	Y	Y	N	N	N	Y
9.	Y	N	Y	Y	Y	N
10.	Y	N	Y	Y	N	N
11.	Y	N	Y	N	Y	N
12.	Y	N	Y	N	N	N
13.	Y	N	N	Y	Y	Y
14.	Y	N	N	Y	N	N
15.	Y	N	N	N	Y	Y
...						
32.	N	N	N	N	N	N

Example diagnosis

Using the table, we look for minimal sets of literals.

For example, from line (5), we have that

$$\text{Ab}(b1) \wedge \text{Ab}(b2) \wedge \neg \text{Ab}(a1) \wedge \text{Ab}(a2) \wedge \text{Ab}(o1)$$

entails the observations. However, lines (5), (7), (13) and (15) together lead us to a smaller set of literals (the first explanation below).

The explanations are

1. $\text{Ab}(b1) \wedge \neg \text{Ab}(a1) \wedge \text{Ab}(o1)$
2. $\text{Ab}(b1) \wedge \neg \text{Ab}(a1) \wedge \neg \text{Ab}(a2)$
3. $\text{Ab}(b2) \wedge \neg \text{Ab}(a1) \wedge \text{Ab}(o1)$

Note: not all components are mentioned since for these settings, get the same observations whether or not they are working

but for this fault model only

Can narrow down diagnosis by looking at a number of different settings
differential diagnosis